Machine Learning 2

Assignment 2

Junyoung Jung

b) Visualize at least one image for each class. You may need to look into how dataset is implemented in PyTorch.

A screenshot of a cell phone

Description automatically generated

I used the matplotlib library to plot the images. To show all the images, I had to create a subplot and grid them accordingly. To access one image per class, I created a “checkbox” list that is to mark the classes that have appeared. Then, I iterated through the dataset and continued if the label has appeared.

c) Split the trainset into training set and validation set with 90% : 10% ratio. Implement dataloaders for CIFAR10.

To separate the training set and the validation set, I, first, measured the length of the training set and divided them in to two parts, each being 90 to 10 ratio. Then, I used the random split method in torch.utils.data module to randomly split the data in the training set to training and validating data. After inserting the splitted data in to the training and validation set, I put them in to the dataloader for iteration purpose. As the datalodaer has “batch” parameter to utilize the batch size, the option is kept open until the batch is required in latter performance.

d) Choose any two classes. Then, make a SVM classifier (implement a loss function yourself.

Do not use PyTorch implementations of loss functions.) and its training/validation/evaluation

code to perform binary classification between those two classes.

I have chosen label 3 and 7 for classification.

e) Train for 10 epochs with batch size 64.

With SGD(Stochastic Gradient Descent) accommodated as the optimizer, batch size 64, learning rate 0.01, the training loss seemed to fluctuate very aggressively. The loss were in between 0.3 to 2.0 for the first epoch and 0.4 to 1.9 by the seventh epoch. This showed that the gradient was calculating at a very fast tempo.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer

Description automatically generated

With no training at all, the validation and testing result comes out as 10.2% accuracy with average loss of 0.205 and 10.0% accuracy with 0.201 average loss.

A screenshot of a computer error

Description automatically generated

After training with batch size of 64, C of hinge loss 0.1, learning rate of 0.005, and 10 epochs,

The validation and test error came out as

A screenshot of a computer

Description automatically generated

As the loss was fluctuating throughout the training, I have tried to decrease the learning rate to 0.0005 first. The result came out as, validation of 19.48% correctness and test being 20%.

A screenshot of a computer error

Description automatically generated

As the learning rate did not influence critically to the model’s accuracy, I have adjusted the value of C in the hingeloss to 10.

f) Perform data normalization. You may need to look into how to use datasets in PyTorch.

In order to readjust the data to be normalized, I have used the transforms.Compose() method where there is transforms.ToTensor() and transforms.Normalize() functions. To normalize the data, I used the transforms.Normalize() function and adjusted the data to (0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261) – which was the optimal ratio indicated according to “dlmacedo” at <https://github.com/kuangliu/pytorch-cifar/issues/19>

g) Again, train for 10 epochs with batch size 64 after data normalization. Write down your

observations.

After 10 epochs with batch size 64, the result was surprisingly intact. The result still viewed

h) What are the hyperparameters you can tune?

The hyperparameters are the learning rate, the epoch, value of C.

i) Try to obtain find optimal hyperparameters.

The final optimal hyperparameter was C = 0.01, Epoch = 10, Learning Rate = 0.01

j) What is the final test accuracy?

A screen shot of a computer error

Description automatically generatedA screenshot of a computer error

Description automatically generated